U.S. Patent No. 7,519,814 ("'814 Patent")

Accused Instrumentalities: Google's "Migrate to Containers," and all versions and variations thereof since the issuance of the asserted patent.

Claim 1

Claim 1	Accused Instrumentalities
[1pre] 1. In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications are executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising:	To the extent the preamble is limiting, Google practices, through the Accused Instrumentalities, in a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications are executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, as claimed. *See* claim limitations below. *See also, e.g.*: Use Migrate to Containers to modernize traditional applications away from virtual machine (VM) instances and into native containers that run on Google Kubernetes Engine (GKE), Anthos clusters, or Cloud Run platform. You can migrate workloads from VMs that run on VMware or Compute Engine, giving you the flexibility to containerize your existing workloads with ease. https://cloud.google.com/migrate/containers/docs/getting-started, Last accessed on June 17, 2023 Given that, using tools like Migrate to Containers is a uniquely smart, efficient way to modernize traditional applications away from virtual machines and into native containers. Our unique automation approach extracts critical application elements from a VM so you can easily insert those elements into containers running on Google Kubernetes Engine (GKE), **without** artifacts like guest OS layers that VMs need but that are unnecessary for containers. https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities

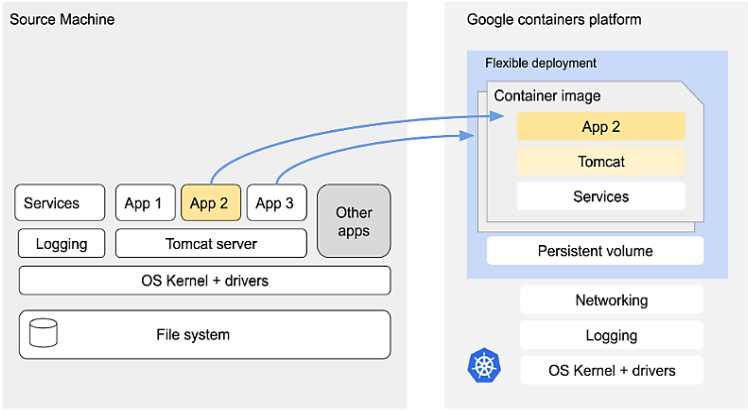
Migrate to Containers supports migrations of VMs to containers on Google Kubernetes Engine on the 64-bit Linux operating systems listed in the following table.

OS	Compute Engine	VMware
CentOS	6.0, 7.0, 7.0 UEFI, 8.0	6.7, 6.9, 7.6
Debian	7.0, 8.0, 9.0, 10.0	9.4, 9.6
RHEL	6.0, 7.0, 7.0 UEFI, 7.4 SAP, 7.6 SAP, 8.0	6.5, 7.5, 7.6, 8.3
SUSE	12, 12 SP3 SAP, 12 SP4 SAP, 15, 15 SAP, 15 SP1 SAP	12 SP2, 12 SP3, 12 SP4, 15
Ubuntu	12 LTS, 14 LTS, 16 LTS, 16 LTS minimal, 18 LTS, 18 LTS minimal, 18 LTS UEFI, 19.04, 19.04 minimal	12.04.5 LTS, 14.04 LTS, 16.04 LTS, 18.04.10 LTS

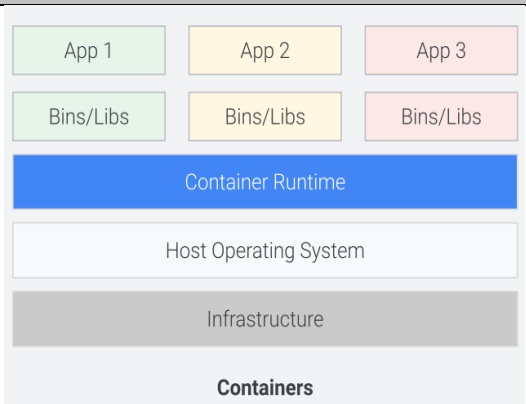
https://cloud.google.com/migrate/containers/docs/compatible-os-versions, Last accessed on June 05, 2023

Containers can run virtually anywhere, greatly easing development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or on physical servers; on a developer's machine or in data centers on-premises; and of course, in the public cloud.

https://cloud.google.com/learn/what-are-containers, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
	A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023  https://cloud.google.com/blog/products/application-modernization/shift-your-apps-to-container-based-workloads-on-the-command-line, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities



https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023

Containers virtualize CPU, memory, storage, and network resources at the operating system level, providing developers with a view of the OS logically isolated from other applications.

https://cloud.google.com/learn/what-are-containers, Last accessed on June 17, 2023

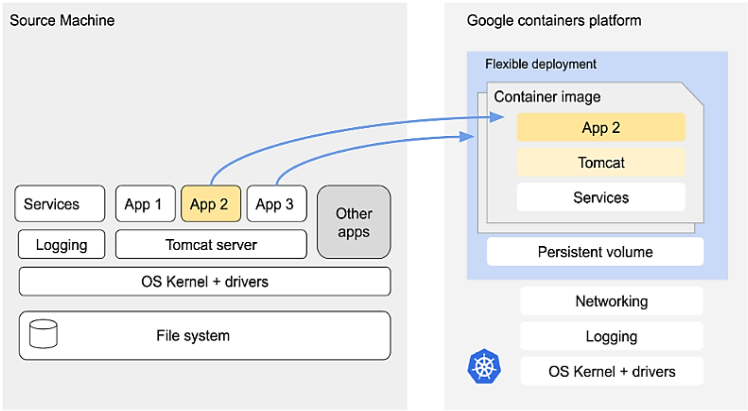
Containers are much more lightweight than VMs

Containers virtualize at the OS level while VMs virtualize at the hardware level

Containers share the OS kernel and use a fraction of the memory VMs require

https://cloud.google.com/learn/what-are-containers, Last accessed on June 17, 2023

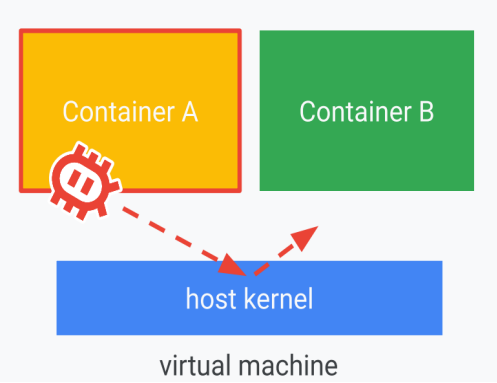
Claim 1	Accused Instrumentalities
	Containers use specific features of the Linux kernel that "trick" individual applications into thinking they're in their own unique environment, even though multiple applications share the same host kernel. (If you're not familiar with the Linux kernel, it's a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests). https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023 The core components of the Linux kernel that are used for containers are **cgroups** — control groups, which define the resources like CPU and memory which are available to a given process — and **namespaces**, which are a way of separating processes by restricting what each process can see, so that system resources "appear" isolated to the process. https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023
[1a] storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers;	The method practiced by Google through the Accused Instrumentalities includes a step of storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers. *See, e.g.*: There are several storage options for applications running on Google Kubernetes Engine (GKE). The choices vary in terms of Volumes are a storage unit accessible to containers in a Pod. Some volume types are backed by ephemeral storage. Ephemeral storage types (for example, emptyDir) do not persist after the Pod ceases to exist. These types are useful for scratch space for applications. You can manage your local ephemeral storage resources as you do your CPU and memory resources. Other volume types are backed by durable storage.

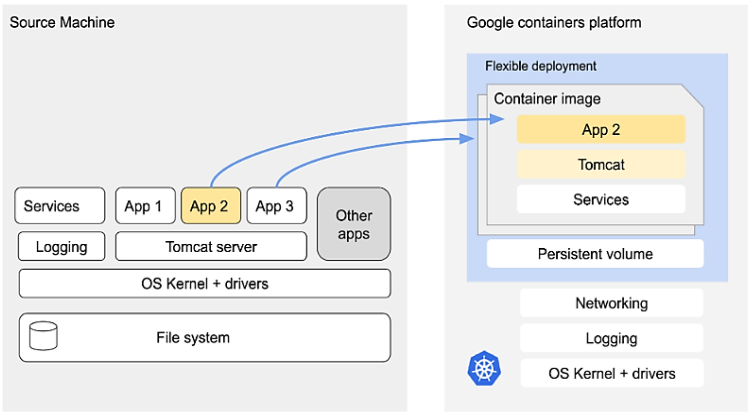
Claim 1	Accused Instrumentalities
	https://cloud.google.com/kubernetes-engine/docs/concepts/storage-overview, Last accessed on June 17, 2023 At its core, a volume is a directory, possibly with some data in it, which is accessible to the containers in a pod. How that directory comes to be, the `.spec.containers[*].volumeMounts`. A process in a container sees a filesystem view composed from the initial contents of the container image, plus volumes (if defined) mounted inside the container. The process sees a root filesystem that initially matches the contents of the container image. Any writes to within that filesystem hierarchy, if allowed, affect what that process views when it performs a subsequent filesystem access. Volumes mount at the specified paths within the image. For each container defined within a Pod, you must independently specify where to mount each volume that the container uses. https://kubernetes.io/docs/concepts/storage/volumes/, Last accessed on June 17, 2023  https://cloud.google.com/blog/products/application-modernization/shift-your-apps-to-container-based-workloads-on-the-command-line, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
	A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities. workloads onto each server. As such, the architecture of containers means that they're deployed with multiple containers sharing the same kernel. https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023 Containers are lightweight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services. https://cloud.google.com/learn/what-are-containers, Last accessed on June 17, 2023
[1b] wherein the set of associated system files are compatible with a local kernel	In the method practiced by Google through the Accused Instrumentalities, the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems.

Claim 1	Accused Instrumentalities
of at least some of the plurality of different operating systems,	*See, e.g.*: A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities. Containers use specific features of the Linux kernel that "trick" individual applications into thinking they're in their own unique environment, even though multiple applications share the same host kernel. (If you're not familiar with the Linux kernel, it's a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests). https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
	Containers can run virtually anywhere, greatly easing development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or on physical servers; on a developer's machine or in data centers on-premises; and of course, in the public cloud. https://cloud.google.com/learn/what-are-containers, Last accessed on June 17, 2023
[1c] the containers of application software excluding a kernel,	In the method practiced by Google through the Accused Instrumentalities, the containers of application software exclude a kernel. *See, e.g.*: • **Higher utilization and density**, leveraging automatic bin-packing and auto-scaling capabilities, Kubernetes places containers optimally in nodes based on required resources while scaling as needed, without impairing availability. In addition, unlike VMs, all containers on a single node share one copy of the operating system and don't each require their own OS image and vCPU, resulting in a much smaller memory footprint and CPU needs. This means more workloads running on fewer compute resources. https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
	workloads onto each server. As such, the architecture of containers means that they're deployed with multiple containers sharing the same kernel.  https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023
[1d] wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server,	In the method practiced by Google through the Accused Instrumentalities, some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server. *See, e.g.*:

Claim 1	Accused Instrumentalities
	One of the primary reasons to adopt containers is for your applications to be decoupled from the underlying environment and support higher resource utilization by "bin packing" multiple workloads onto each server. As such, the architecture of containers means that they're deployed with multiple containers sharing the same kernel. The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities. https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023  https://cloud.google.com/blog/products/application-modernization/shift-your-apps-to-container-based-workloads-on-the-command-line, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
[1e] wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server,	In the method practiced by Google through the Accused Instrumentalities, said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server. *See, e.g.*: One of the primary reasons to adopt containers is for your applications to be decoupled from the underlying environment and support higher resource utilization by "bin packing" multiple workloads onto each server. As such, the architecture of containers means that they're deployed with multiple containers sharing the same kernel. The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities. https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023
[1f] and wherein the application software cannot be shared between the plurality of secure containers of application software,	In the method practiced by Google through the Accused Instrumentalities, the application software cannot be shared between the plurality of secure containers of application software. *See, e.g.*:

Claim 1	Accused Instrumentalities
	Containers use specific features of the Linux kernel that "trick" individual applications into thinking they're in their own unique environment, even though multiple applications share the same host kernel. (If you're not familiar with the Linux kernel, it's a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests). https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023 The core components of the Linux kernel that are used for containers are **cgroups** — control groups, which define the resources like CPU and memory which are available to a given process — and **namespaces**, which are a way of separating processes by restricting what each process can see, so that system resources "appear" isolated to the process. https://services.google.com/fh/files/misc/why_container_security_matters.pdf, Last accessed on June 17, 2023 reason. Furthermore, files within a container are inaccessible to other containers running in the same Pod. The Kubernetes https://cloud.google.com/kubernetes-engine/docs/concepts/volumes, Last accessed on June 17, 2023 A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled. In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host. The shared context of a Pod is a set of Linux namespaces, cgroups, and potentially other facets of isolation - the same things that isolate a container. Within a Pod's context, the individual applications may have further sub-isolations applied. https://kubernetes.io/docs/concepts/workloads/pods/, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
	ranges can access. GKE Sandbox for the Standard mode of operation provides a second layer of defense between containerized workloads on GKE for enhanced workload security. GKE https://cloud.google.com/kubernetes-engine#section-2, Last accessed on June 17, 2023
[1g] and wherein each of the containers has a unique root file system that is different from an operating system's root file system.	In the method practiced by Google through the Accused Instrumentalities, each of the containers has a unique root file system that is different from an operating system's root file system. *See, e.g.*: The original purpose of the cgroup, `chroot`, and namespace facilities in the kernel was to protect applications from noisy, nosey, and messy neighbors. Combining these with container images created an abstraction that also isolates applications from the (heterogeneous) operating systems on which they run. This decoupling of image and OS makes it possible to provide the same deployment environment in both development and production, which, in turn, improves deployment reliability and speeds up development by reducing inconsistencies and friction. "Borg, Omega, and, Kubernetes," https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44843.pdf, Last accessed on June 17, 2023

Claim 1	Accused Instrumentalities
	In Docker and Kubernetes, the container's root filesystem (rootfs) is based on the filesystem packaged with the image. The image's filesystem is immutable. Any change a container makes to the rootfs is stored separately and is destroyed with the container. This way, the image's filesystem https://opensource.googleblog.com/2023/04/gvisor-improves-performance-with-root-filesystem-overlay.html, Last accessed on June 17, 2023 To use a volume, specify the volumes to provide for the Pod in `.spec.volumes` and declare where to mount those volumes into containers in `.spec.containers[*].volumeMounts`. A process in a container sees a filesystem view composed from the initial contents of the container image, plus volumes (if defined) mounted inside the container. The process sees a root filesystem that initially matches the contents of the container image. Any writes to within that filesystem hierarchy, if allowed, affect what that process views when it performs a subsequent filesystem access. Volumes mount at the specified paths within the image. For each container defined within a Pod, you must independently specify where to mount each volume that the container uses. https://kubernetes.io/docs/concepts/storage/volumes/, Last accessed on June 17, 2023